Національний технічний університет України «КПІ»

Кафедра автоматизованих систем обробки інформації і управління

Лабораторна робота №3

з дисципліни «[Аналіз даних в інформаційних управляючих системах](https://do.ipo.kpi.ua/course/view.php?id=5470)»

на тему: «[ДЕРЕВА ВИРІШАЛЬНИХ ПРАВИЛ](https://do.ipo.kpi.ua/mod/resource/view.php?id=149461)»

Виконав:

студент групи ІС-23

Шимків М.В.

Викладач:

Гавриленко О.В

Київ 2024

**Мета роботи**: вивчити основні методи побудови дерев вирішальних правил,

навчитися використовувати спеціалізовані програмні засоби для побудови дерев вирішальних правил.

**Завдання до роботи**

1. Ознайомитися з конспектом лекцій та рекомендованою літературою, а також додатком до лабораторної роботи, що містить опис програмного забезпечення для

побудови дерев вирішальних правил.

2. Сформувати набір даних для обробки та аналізу.

3. Використовуючи рекомендоване програмне забезпечення здійснити обробку

набору даних з метою побудови дерева вирішальних правил.

4. Використати побудоване дерево для прийняття рішень на конкретному

прикладі.

5. Оформити звіт з роботи.

6. Відповісти на контрольні питання.

**Набір даних:**

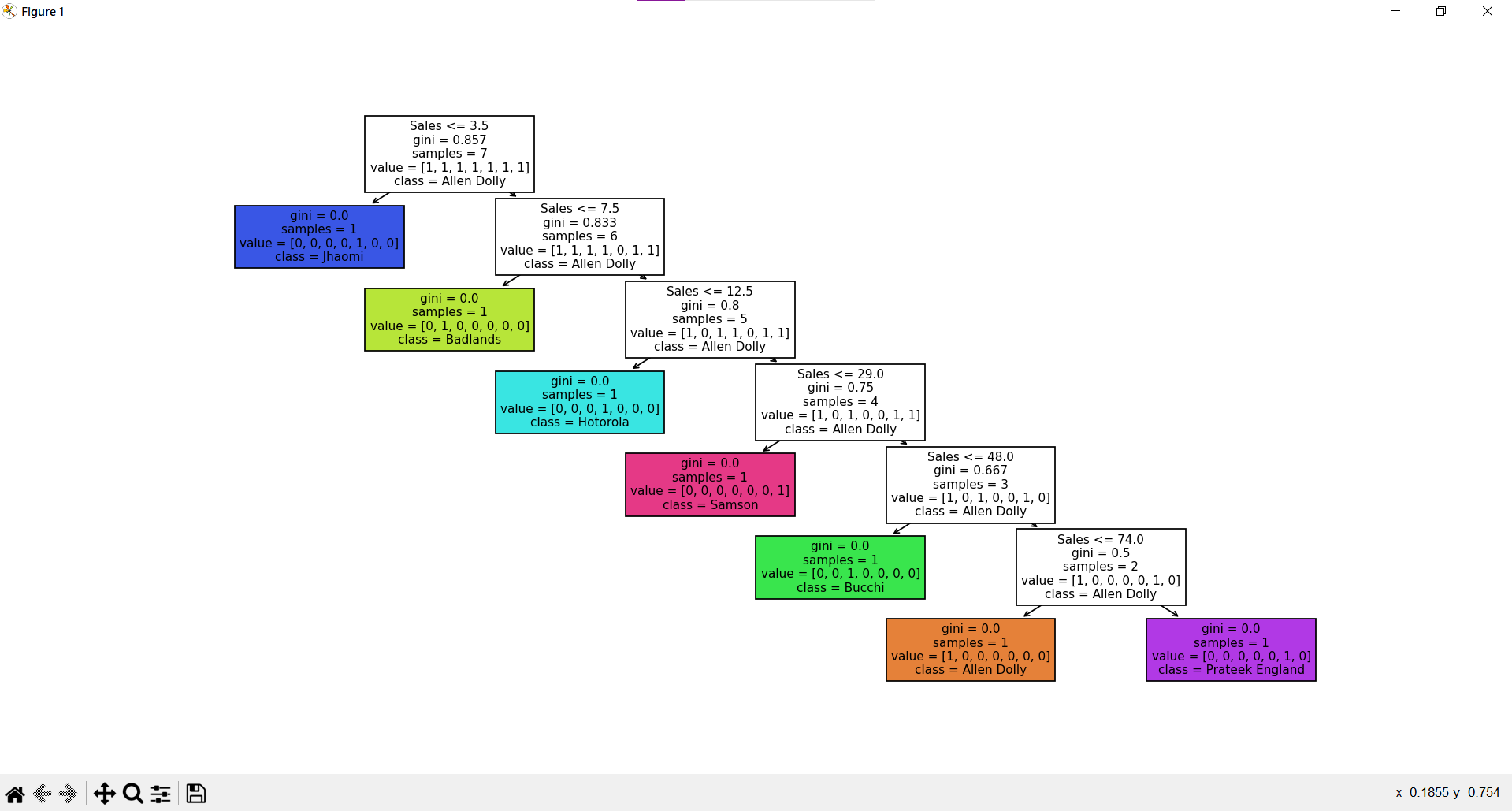
Набір даних я взяв з сервісу kaggle. Ось посилання на набір даних:

<https://www.kaggle.com/datasets/varunraskar/store-sales-data>

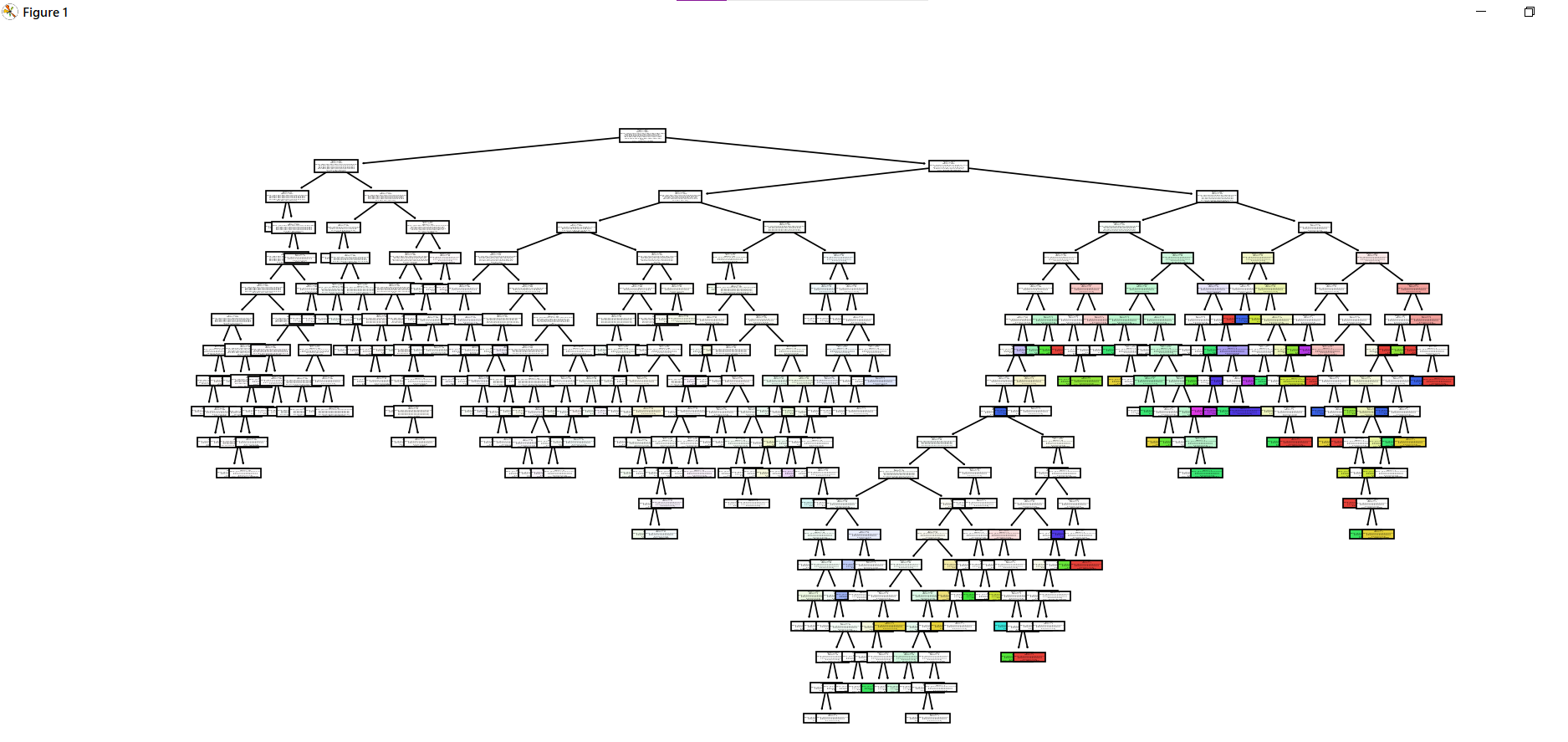
Запуск програми відбувається так:



Побудоване дерево прийняття рішень(на меншій кількості даних):

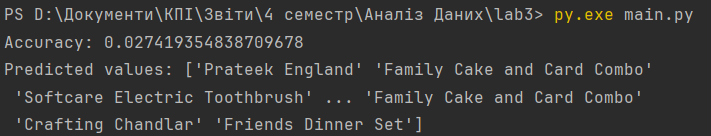


Побудоване дерево прийняття рішень(на більшій кількості даних):



Використаємо побудоване дерево для прийняття рішень на конкретному

прикладі:





**Код програми:**

from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.tree import DecisionTreeClassifier, plot\_tree  
  
# Метод для побудови дерева вирішальних правил  
def treefit(X, y):  
 clf = DecisionTreeClassifier()  
 clf.fit(X, y)  
 return clf  
  
# Метод для обрізання дерева за рівнем або кількістю вузлів  
def treeprune(t, criteria, value):  
 if criteria == 'level':  
 t.max\_depth = value  
 elif criteria == 'nodes':  
 t.max\_leaf\_nodes = value  
  
# Метод для відображення дерева  
def treedisp(t):  
 print(t.tree\_)  
  
# Метод для тестування дерева на тестовому наборі  
def treetest(t, X\_test, y\_test):  
 y\_pred = t.predict(X\_test)  
 return accuracy\_score(y\_test, y\_pred)  
  
# Метод для оцінки значень вихідного параметру Ycalc за допомогою дерева t та масиву незалежних змінних X  
def treeval(t, X):  
 Ycalc = t.predict(X)  
 return Ycalc  
  
# Завантаження набору даних з CSV файлу  
def load\_data\_from\_csv(file\_path):  
 df = pd.read\_csv(file\_path)  
 return df  
  
# Пункт 2: Завантаження набору даних  
data\_path = 'Sales.csv'  
df = load\_data\_from\_csv(data\_path)  
  
# Розділення набору даних на ознаки та цільову змінну  
X = df[['Sales']]  
y = df['Product Name']  
  
# Розділення набору даних на тренувальний та тестовий  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Пункт 1: Побудова дерева вирішальних правил  
clf = treefit(X\_train, y\_train)  
  
# Пункт 3: Відображення дерева  
plt.figure(figsize=(10,10))  
plot\_tree(clf, filled=True, feature\_names=X.columns, class\_names=clf.classes\_)  
plt.show()  
  
# Пункт 4: Тестування дерева на тестовому наборі  
accuracy = treetest(clf, X\_test, y\_test)  
print("Accuracy:", accuracy)  
  
# Пункт 5: Розрахунок значень вихідного параметру Ycalc за допомогою дерева t  
Ycalc = treeval(clf, X)  
print("Predicted values:", Ycalc)  
  
# Приклад тестового запису  
test\_sample = [[500]] # Припустимо, ми маємо тестовий запис, де продажі дорівнюють 500  
  
# Використання побудованого дерева для прийняття рішення на конкретному прикладі  
predicted\_class = clf.predict(test\_sample)  
print("Predicted class:", predicted\_class)

**Контрольні питання:**

1. Дерево вирішальних правил - це модель машинного навчання, яка використовується для класифікації або регресії. Вони представляють собою деревоподібну структуру, де кожен вузол представляє тест на певному атрибуті, кожне ребро представляє результат тесту, а кожен лист відповідає мітці класу або прогнозу.
   * Об'єкт: це екземпляр даних або вхід, який ми хочемо класифікувати або прогнозувати.
   * Атрибут: це характеристика об'єкта, яка використовується для прийняття рішення в дереві.
   * Мітка класу: це категорія або клас, який ми хочемо присвоїти об'єкту.
   * Вузол: це точка в дереві, де відбувається перевірка значення атрибута.
   * Лист: це кінцевий вузол в дереві, який відповідає мітці класу або прогнозу.
   * Перевірка: це умова або правило, яке визначає, як об'єкт повинен бути розбитий на частини в дереві.
2. Основні класи задач, до яких можуть бути застосовані дерева вирішальних правил, включають класифікацію, регресію і прогнозування.
3. Побудова дерева рішень відбувається шляхом рекурсивного розбиття набору даних на підмножини на основі значень атрибутів. Для цього можуть використовуватися різні алгоритми, такі як CART (Classification and Regression Trees) або C4.5.
4. Процес навчання з учителем полягає в тому, що модель отримує набір навчальних даних, який включає вхідні об'єкти та відповідні мітки класів або цільові значення, і навчається прогнозувати ці мітки або значення.
5. CART (Classification and Regression Trees) та C4.5 - два основних методи побудови дерев вирішальних правил. CART здатний працювати як з класифікацією, так і з регресією, тоді як C4.5 призначений для класифікації.
6. Принцип роботи "жадібних" алгоритмів полягає в тому, що на кожному кроці вони вибирають найкращий доступний варіант без зазирання в майбутнє. Це означає, що алгоритм приймає найкраще рішення на кожному кроці, навіть якщо це може призвести до неоптимального рішення в цілому.
7. При побудові дерев рішень зазвичай приділяється увага таким аспектам, як вибір оптимального атрибута для розбиття, зупинка росту дерева, уникнення перенавчання та оптимізація шляху розбиття.
8. Правило відбору ознаки для розбиття полягає в тому, що ми шукаємо атрибут, який найкращим чином розділяє набір даних на дві або більше підмножини. Загальне правило для відбору атрибуту - обчислити певний критерій (наприклад, ентропію або джині) для кожного атрибута і вибрати той, який мінімізує цей критерій після розбиття.
9. Критерії оцінки якості розбиття множини на класи включають ентропію, джині і помилку класифікації. Кожен з цих критеріїв вимагає максимізації інформаційної чистоти або мінімізації невпевненості після розбиття.
10. Правило зупину визначає, коли рекурсивний процес побудови дерева рішень повинен бути припинений. Це може бути досягненням певної глибини дерева, досягненням мінімальної кількості об'єктів у вузлі або іншими критеріями.
11. Правило відсіку використовується для уникнення перенавчання та покращення загальної загальноїздатності моделі. Воно може бути застосоване для обмеження глибини дерева, мінімальної кількості об'єктів у вузлі або інших параметрів.
12. Точність для дерева вирішальних правил визначається як відсоток правильних передбачень, зроблених моделлю на навчальному або тестовому наборі даних. Помилка розпізнавання - це відсоток неправильних передбачень.
13. Для добування правил з дерева вирішальних правил потрібно пройти кожним шляхом від кореня до листя та сформувати умови на кожному шляху, які призводять до класифікації об'єктів або прогнозування значень.
14. Метод C4.5 має вимоги до структури даних, що включають наявність атрибутів зі значеннями, які можна категоризувати, та класифікаційні або регресійні мітки.
15. Алгоритм побудови дерева вирішальних правил за допомогою методу C4.5 полягає у виборі оптимального атрибуту для розбиття на кожному кроці, враховуючи критерій інформаційної вигоди.
16. Критерій вибору атрибуту в методі C4.5 визначається за допомогою критерію інформаційної вигоди, який враховує розподіл класів вузла перед і після розбиття.
17. Вузол у методі C4.5 помічається як лист, коли всі об'єкти вузла належать до одного класу, а розв'язком листа обирається цей клас.
18. Ентропія досягає свого максимуму, коли всі класи у вузлі рівномірно розподілені, і досягає свого мінімуму, коли всі об'єкти вузла належать до одного класу.
19. Поріг для порівняння значень атрибуту обирається тоді, коли атрибут має неперервний тип даних, і потрібно розбити значення на дві або більше категорії.
20. Класифікація нових об'єктів у дереві вирішальних правил полягає у проходженні через дерево від кореня до листя, починаючи з кореня.
21. Покращений критерій розбиття зазвичай використовується для зменшення перенавчання шляхом врахування розміру вузла та кількості класів.
22. Для зменшення ймовірності створення вузлів та листя, які містять незначну кількість об'єктів, може використовуватися евристичне правило, що обмежує розбиття, якщо воно не призведе до суттєвого збільшення інформаційної вигоди.
23. Процедура роботи з пропущеними даними включає їх обробку шляхом виключення з розгляду або заповнення відсутніх значень, наприклад, середніми або медіанами.
24. У випадку відсутності значення певного атрибуту об'єкту, що класифікується, він може бути віднесений до одного з піддерев за відсутністю відповідного правила, або може бути використана інша стратегія обробки відсутніх даних.
25. Переваги використання дерев вирішальних правил
26. При використанні методу ID3 для побудови дерева вирішальних правил застосовується напрямок, який максимізує інформаційний приріст на кожному кроці.
27. Послідовність побудови дерева вирішальних правил за допомогою методу ID3 включає наступні кроки:
28. Початковий набір даних.
29. Вибір кореневого вузла шляхом вибору властивості з найбільшим інформаційним приростом.
30. Розділення набору даних на піднабори на основі значень обраної властивості.
31. Повторення кроків 2-3 для кожного новоутвореного піднабору рекурсивно до досягнення критерію зупинки, такого як досягнення певної глибини дерева або досягнення певної мінімальної кількості зразків у вузлі.

Ця послідовність кроків дозволяє рекурсивно побудувати дерево вирішальних правил, вибираючи найбільш інформативні властивості для розділення даних на кожному кроці.

1. Вибір властивості на основі теорії інформації полягає в оцінці інформаційного приросту, який отримується при розділенні набору даних за певною властивістю. Цей приріст інформації вимірюється за допомогою метрики, такої як ентропія або критерій Джині. Основна ідея полягає в тому, що властивість, яка розділяє набір даних на більш однорідні групи (зменшує невизначеність або "несподіваність" у кожній групі), має більший інформаційний приріст і, отже, є більш важливою для побудови дерева вирішальних правил. Таким чином, вибір властивостей здійснюється шляхом максимізації інформаційного приросту на кожному кроці побудови дерева.
2. У пакеті MATLAB для роботи з деревами вирішальних правил доступні різні функції та можливості, які допомагають у побудові, навчанні та використанні таких моделей. Основні елементи пакету включають:
3. **Внутрішня структура**: Дерева вирішальних правил у MATLAB представлені структурою, що містить інформацію про вузли та розгалуження.
4. **Параметри**: Параметри дерева вирішальних правил можуть включати критерії розділення (наприклад, ентропія або критерій Джині), максимальну глибину дерева, мінімальну кількість прикладів для поділу тощо. Ці параметри можна налаштовувати для оптимізації моделі під конкретні завдання.
5. **Основні змінні**: Зазвичай включають навчальні дані та параметри моделі. Навчальні дані містять вхідні характеристики та відповідні цільові значення, які використовуються для навчання моделі. Параметри моделі визначаються під час навчання та використовуються для налаштування дерева.
6. **Методи**: Пакет MATLAB містить різні методи для роботи з деревами вирішальних правил, такі як:
   * fit: для навчання моделі на навчальних даних.
   * predict: для прогнозування цільових значень для нових вхідних даних на основі навченої моделі.
   * view: для візуалізації структури дерева.
   * Інші методи для аналізу та оцінки моделі.

**Висновок:**

Отже, завдяки цій лабораторній роботі я отримав глибше розуміння теорії та практичного застосування дерев вирішальних правил. Я здобув навички з обробки даних, побудови дерев та прийняття рішень на їх основі. Також зрозумів принципи роботи "жадібних" алгоритмів та різницю між методами CART та C4.5. Вважаю, що ця робота допомогла мені покращити мої навички в аналізі даних та вирішенні завдань з використанням машинного навчання.